REMARKS

Applicant is in receipt of the Office Action mailed February 7, 2007. Claims 1-23 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Section 103 Rejections

Claims 1 and 18-21 were rejected under 35 U.S.C. 103(a) as being unpatentable over PerfectXML (http://www.perfectxml.com/nr/aspnetdebug.pdf) in view of Michelman et al (US Pat. 6,907,580, "Michelman"). Applicant respectfully disagrees.

The Office Action asserts that Microsoft Corporation's Visual Studio .NET debugging environment, as described in PerfectXML, and Michelman, include all the features and limitations of claim 1.

Claim 1 recites

1. A memory medium which stores program instructions implementing a graphical user interface (GUI) for debugging a program, wherein, during execution of the program, the program instructions are executable by a processor to perform:

displaying source code for the program in a first GUI element;

receiving first user input to the first GUI element indicating an expression in the source code;

displaying a value of the expression in a tooltip in response to said first user input;

receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression; and

setting the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression.

The Office Action asserts that Michelman is directed to a "graphical user interface (GUI) for debugging a program". Applicant respectfully disagrees, noting that Michelman nowhere mentions or even hints at debugging a program. As Michelman's

Abstract makes clear, Michelman is directed to a moveable displayed user interface element that can be used to select an item displayed on a user interface, after which an area of the moveable displayed user interface element can be activated to perform an operation on the selected item or items. More particularly, Michelman's UI elements are described as interfaces for performing specified operations on items upon activation of an area *on the user interface element*, e.g., by pressing a button, etc. Additionally, Michelman's UI element is moved around on the screen to select the items to be operated on, e.g., by the user dragging the element around with a mouse/cursor. This is in direct contrast with a tooltip, which, as is well-known to those of skill in the art of GUIs, is a GUI element that is displayed automatically and dynamically, e.g., when the user hovers a cursor over an item, and is *not* dragged around on the screen via a pointing device.

Applicant respectfully submits that there are numerous features not taught or suggested by the cited art.

For example, nowhere does the cited art teach or suggest **receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression**, as recited in claim 1.

The Office Action admits that PerfectXML fails to teach or suggest a tooltip that accepts user input, but asserts that Michelman teaches "receiving first user input to the first GUI element indicating an expression in the source code", citing col.2:66-67, and (col.3?), lines 1-3, stating "wherein the user places a cursor over an object such that of an expression in a debug application [sic]". Nowhere does Michelman disclose debugging at all, nor "the user placing a cursor over an expression in a debug application". The Examiner appears to be attempting to insert specifics from Applicant's claimed invention into Michelman's system without any supporting evidence that Michelman discloses these features, which is improper. Applicant notes that Michelman fails to disclose placing a cursor over any type of expression in source code at all, but rather discusses dragging a GUI element around on a screen using a cursor to place the GUI element in specific regions, and so the GUI element is clearly not the cursor.

The cited col.2:66-col.3:3 reads:

> FIGS. 1A-1C show a variety of exemplary moveable displayed user interface elements that can be used to select a set of one or more items and perform an operation on the selected items upon activation of an area on the user interface element.

Applicant respectfully submits that the cited text, and Michelman in general, fails to teach or suggest the claimed feature, and further fails to disclose the asserted "receiving first user input to the first GUI element indicating an expression in the source code". In fact, Michelman nowhere discusses displaying "source code" in a first GUI element, and in fact fails to mention "source code" at all, and is not directed to debugging programs. Nor does Michelman describe or even mention tooltips, and more specifically fails to describe the tooltip edit functionality claimed.

Applicant respectfully notes that the first GUI element of claim 1 is distinct from the claimed tooltip, and so the asserted "receiving first user input to the first GUI element indicating an expression in the source code" is not germane to the claimed limitation of **receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression**. For example, according to Applicant's Specification and Figures, an exemplary first GUI element in which source code may be displayed is a source code debugging window. Applicant further respectfully notes that Michelman nowhere teaches receiving user input indicating (e.g., selecting) an expression in displayed source code. In fact, Michelman never even mentions "expression" or "source code" at all. Rather, per Michelman, Michelman's system is directed to performing operations on selected items, and provides numerous examples of such operations. For example, col.5:14-19 reads:

> …if the items represent documents, the operations can be "edit," "delete," or "rename" operations. If the items represent merchandise, the operations can be "get information" or "buy" operations. If the items are columns in a table, the operations can be "modify," "rename," "delete," "format," "add," or "move" operations.

Clearly, the operations contemplated by Michelman are directed to such operations as editing/deleting/renaming files, buying items, retrieving information, and editing or reformatting tables. Nowhere does Michelman discuss changing the value of an expression in a program, nor, more specifically, changing the value of such an expression within a tooltip during program execution to debug the program.

In asserting that Michelman discloses **receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression**, the Office Action cites col.4:31, which reads:

> In some cases, it may be desirable to allow editing of the information displayed in the text box 136 (e.g., to change a selected item's name).

Applicant respectfully notes that the cited text is directed to editing information displayed in a text box of a GUI element that the user moves (via a cursor) over an item to be operation on. The Examiner attempts to characterize Michelman's UI element as a "tool tip", but Michelman never mentions a tool tip, nor editing within a tool tip. Applicant respectfully submits that it is improper for the Examiner to attempt to redefine well-known terms of art in contradiction to their established meanings. The cited GUI element of Michelman is *not* a tooltip, is nowhere described as a tooltip, and does not operate as a tooltip operates. Nor is the information displayed in the text box ever described or characterized as a value of an expression in the source code of a program during the program's execution.

Thus, Applicant submits that the cited text (and Michelman in general) fails to teach or suggest **receiving second user input to the tooltip modifying the displayed value**.

The Office Action asserts that Michelman discloses **thereby specifying a new value for the expression**, col.9:21-24.

As noted above, col.4:31 mentions editing information displayed in a text box, e.g., to change a selected item's name, but does not mention or even hint at specifying a new value for an expression in program source code during execution of the program.

Col.9:21-24 reads:

> When the user interface element is moved over one of the targets, a function is invoked to pass information about the associated region (and thus a reference to the target) to a software object associated with the user interface element. At such time, the software object associated with the user interface element can take any of a number of actions, including making a notation of the target over which the user interface element has been positioned.

As may be seen, the cited text simply appears to indicate that a notation of the target over which the user interface element has been positioned may be made, which is not at all the same as specifying a new value for an expression in program source code during execution of the program.

Thus, PerfectXML and Michelman in combination fail to teach this feature of claim 1.

In Office Action asserts that PerfectXML discloses **setting the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression**, citing PerfectXML, p.108, which describes a standard watch window in a debugging environment whereby the user may change the value of a watched variable, control, or other object. The Examiner then asserts that it would have been obvious to combine the teachings of ETSU with PerfectXML. Applicant assumes that the reference to ETSU was unintentional.

The Examiner further asserts that it would have been obvious to combine the teachings of PerfectXML with Michelman "because the system and method can be modified to address a variety of other scenarios which that of the watch window replacement [sic]", citing col.7:25, which reads:

> …the system and method can be modified to address a variety of other scenarios.

Applicant respectfully reminds the Examiner that to establish a prima facie obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. In re Bond, 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990).

In addition, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular . . .. Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." In re Dembiczak, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999). The art must fairly teach or suggest to one to make the specific combination as claimed. That one achieves an improved result by making such a combination is no more than hindsight without an initial suggestion to make the combination.

The citation from Michelman does not indicate any desire for or benefit of combining Michelman with PerfectXML. Applicant respectfully submits that the Examiner has in no way provided a proper motivation to combine from the references that is clear and particular, but rather has simply attempted to produce Applicant's claimed invention by assembling selected portions of the cited art using Applicant's claim as a blueprint, without any proper teaching or suggestion from the references. Thus, PerfectXML and Michelman are not available in combination to make a prima facie case of obviousness.

Additionally, Applicant respectfully submits that Michelman is non-analogous art, since Michelman nowhere discusses or even hints at debugging a program. Applicant respectfully submits that one of skill in the art would not be led to examine Michelman in researching the debugging arts, nor in researching tooltip functionality, since Michelman is directed to neither.

Moreover, even were PerfectXML and Michelman properly combinable, which Applicant argues they are not, the resulting combination would still not produce Applicant's invention as represented in claim 1, as discussed at length above. For example, as one of skill in the art would readily understand, a watch window is not a tooltip. Nor is Michalman's UI element a tooltip.

Moreover, Applicant respectfully submits that one of skill in the programming arts would readily understand that modifying the value of a program expression during execution and debugging of a program, and the program continuing execution in accordance with the new value of the expression is technically much more involved than simply modifying a property of an item or editing a document. For example, in the former, the user interacts with the source code, i.e., selecting the expression and changing the value of the expression, and the new value must be substituted in or inserted into the executing program (the executable), which is distinct from the source code, after which the program (executable) continues to execute, but with the new value. This is in contrast with simply changing a value in a data structure, e.g., in a file. Nothing in PerfectXML or Michelman indicates or describes, or even hints at, such functionality.

Thus, even in combination, the cited references do not, and cannot, produce a tooltip as claimed. Thus, Applicant respectfully submits that PerfectXML and Michelman, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 1.

Thus, for at least the reasons provided above, Applicant submits that claim 1 and those claims dependent therefrom are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claims 19 and 20 include similar limitations as claim 1, and so the above arguments apply with equal force to these claims. Thus, for at least the reasons provided above, claim 19 and 20, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Claim 21 also includes many of the novel limitations as claim 1, and so the relevant arguments presented above also apply to this claim; however, instead of displaying the value of the expression and receiving second user input to a tooltip, claim 21 recites **displaying the value of the expression in a window in response to said first user input, wherein the window is operable to display a value of the indicated expression, wherein the window does not include window title bars or menus,** and **receiving second user input to the window modifying the displayed value, thereby specifying a new value for the expression**.

Applicant respectfully submits that the recited window of this claim is distinct from standard GUI windows, such as those discloses in PerfectXML, that include window title bars and/or menus, in that the window of claim 21 specifically includes neither. Thus, PerfectXML and Michelman , taken singly or in combination, fail to teach or suggest all the features and limitations of claim 21.

Thus, for at least the reasons provided above, Applicant submits that claim 21 and those claims dependent therefrom are patentably distinct and non-obvious over the cited art, and are thus allowable.

Removal of the section 103 rejection of claims 1 and 18-21 is earnestly requested.

Claims 1-23 were rejected under 35 U.S.C. 103(a) as being unpatentable over PerfectXML (http://www.perfectxml.com/nr/aspnetdebug.pdf) in view of East Tennessee State University (http://csciwww.etsu.edu/blair/Using_Debugger.htm, "ETSU"). Applicant respectfully disagrees.

The Office Action asserts that Microsoft Corporation's Visual Studio .NET debugging environment, as described in PerfectXML and ETSU, discloses all the features and limitations of claim 1. Applicant respectfully notes that in the Response of November 16, 2006, Applicant provided arguments showing that PerfectXML and ETSU fail to disclose all the features and limitations of claim 1, and that the Examiner has not responded to any of Applicant's arguments. Applicant respectfully requests that the arguments be addressed and/or that the case be allowed.

Applicant respectfully directs the Examiner's attention to the arguments directed to PerfectXML presented above.

As argued above and in the previous Response, nowhere do the cited references teach or suggest **receiving second user input to the tooltip modifying the displayed value, thereby specifying a new value for the expression**, as recited in claim 1.

While the cited art (specifically, ETSU) does disclose displaying a variable value (in a tooltip) in response to holding the mouse cursor over the variable name (referred to as "Mouse Over"), nowhere does the cited art disclose receiving user input to the tooltip modifying the displayed value, thereby specifying a new value. In fact, nowhere do PerfectXML or ETSU disclose receiving user input to a tooltip at all.

Similary, nowhere does the cited art teach or suggest **setting the expression in the program to the new value, wherein the program continues execution in accordance with the new value of the expression**, as further recited in claim 1, where, as claimed, the new value was specified by user input to the tooltip.

The Office Action asserts that PerfectXML teaches receiving second user input to the tooltip, citing page 108 of PerfectXML. However, on p.3 of the Office Action, *the Examiner admits that PerfectXML fails to disclose a tool tip accepting user input.* Thus, by the Examiner's own admission, PerfectXML fails to disclose this limitation of claim 1, as noted above.

Moreover, Applicant has reviewed the PerfectXML reference closely, and respectfully submits that the Examiner has mischaracterized the art. For example, Applicant notes that the citation is directed to displaying and changing values in a watch window, clearly described on page 108, here quoted in pertinent part:

> The watch window enables you to type in a specific variable name, control name, or other object and then see what value it contains and what type it is. For now, type in txtEmail.You will see that you can expand this entry into all the control's properties to see what they each contain.
>
> …
>
> The other thing that you can do with the watch window is change the value of a variable. **<u>If you click the property value in the Value column in the watch window, you can enter a new value as you see fit</u>**.

Clearly, this watch window functionality has no relationship to tooltips and "mouse over" functions in the described debugging environment. The Office Action states "that the watch window is accepting user interaction of modifying a displayed expression in the displayed area called a tool tip", and "this watch window is an editable means 'to the tool tip', wherein the tool tip being that of a display area to display information from the watch window [sic]". Applicant notes that the display area of a watch window is not a tool tip, as is clearly understood by those of skill in the art, and further notes that it is improper for the Examiner to attempt to redefine this well-known term of art in contradiction to its accepted meaning. Applicant respectfully submits that those of skill in the art of debugging readily understand that such watch window functionality, well known in the art, is not at all the same as displaying and changing a value of an expression via a tooltip. Nowhere does the cited art describe or even hint at such tooltip functionality. Moreover, the cited p.7 of ETSU clearly demonstrates that tooltip functionality and watch window functionality are separate and distinct, and thus the Examiner's attempt to equate the two is improper and incorrect.

Regarding claim 21, Applicant respectfully submits that while ETSU does disclose displaying a value in a window without window title bars or menus (referred to as "Mouse Over"), nowhere does ETSU mention or describe receiving user input to such a window (or tooltip). Thus, neither PerfectXML nor ETSU discloses this feature.

Additionally, Applicant respectfully submits that neither PerfectXML nor ETSU discloses setting the expression in the program to the new value that was set via the tooltip (claim 1) or window (without a title bar or menus) (claim 21), where the program continues execution in accordance with the new value of the expression, at least for the reason that neither reference discloses receiving user input to a tooltip or window (without a title bar or menus) to modify the displayed value, where the value of the expression is then set to the modified value, and the program continues to execute with the new value.

Applicant further submits that no proper motivation to combine PerfectXML and ETSU has been provided. For example, the motivation to combine suggested by the Examiner is "because PerfectXML and ETSU disclose a brief introduction of the Microsoft Visual C++ debugger, being that of an introduction not all detailed functionality was disclosed thus the combination of references was used because the brief introduction of PerfectXML did not specifically mention a feature/functionality of the Microsoft Visual C++ debugger that ETSU did discloses in detail [sic]", and "Also the motivation to combine is the presence of the watch window which is disclosed in PerfectXML…and ETSU…wherein PerfectXML shows the functionality and touches briefly on what ETSU covers in more detail on the functionality as mentioned above". Applicant respectfully submits that the cited references of PerfectXML and ETSU both disclose features of the Microsoft Visual C++ debugger, but that, as both references make clear, nothing in each reference indicates or even hints at the desirability or usefulness of receiving user input to a tooltip to modify the value of an expression in the program for debugging purposes. Thus, PerfectXML and ETSU are not available in combination to make a prima facie case of obviousness. Moreover, Applicant respectfully submits that not only would one not be motivated to combine the references to produce Applicant's claimed functionality regarding tooltips, but even were the references combined, the resulting combination would still not teach Applicant invention as claimed, since the Microsoft Visual C++ debugger does not include or support these features.

Thus, the cited art fails to teach or suggest all the features and limitations of claim 1 and claim 21.

Thus, for at least the reasons presented above, PerfectXML and ETSU, taken singly or in combination, fail to teach or suggest all the features and limitations of claims 1 and 21, and so claims 1 and 21, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cite art, and are thus allowable.

Independent claims 18, 19, and 20 include similar limitations as claim 1, and so the above arguments apply with equal force to these claims. Thus, for at least the reasons provided above, Applicant respectfully submits that claims 18, 19, and 20, those claims respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Removal of the 103 rejection of claims 1-23 is respectfully requested.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above-referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. The Commissioner is hereby authorized to charge any fees which may be required or credit any overpayment to Meyertons, Hood, Kivlin, Kowert & Goetzel P.C., Deposit Account No. 50-1505/5150-82801/JCH.

Also filed herewith are the following items:

☐ Request for Continued Examination
☐ Terminal Disclaimer
☐ Power of Attorney By Assignee and Revocation of Previous Powers
☐ Notice of Change of Address
☐ Other:

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood, Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: April 12, 2007 JCH/MSW